Published in: Photogrammetrie, Fernerkundung, Geoinformation PFG, 2007. No. 6, pp. 415-428

A Semantic Model of Stairs in Building Collars

JÖRG SCHMITTWILKEN, JENS SAATKAMP, WOLFGANG FÖRSTNER, THOMAS H. KOLBE*, LUTZ PLÜMER, Bonn (*Berlin)

Keywords: 3D city models, stairs, ontology, stochastic attribute grammars, constraint solver

Summary: The automated extraction of high resolution 3D building models from imagery and laser scanner data requires strong models for all features which are observable at a large scale. In this paper we give a semantic model of stairs. They play a prominent role in the transition from buildings to the surrounding terrain or infrastructure. We name the transition area between terrain and building collar, and the focus is on stairs in building collars. Simple and complex stairways are represented by UML class diagrams along with constraints reflecting semantic and functional aspects in OCL. A systematic derivation of an attribute grammar consisting of production and semantic rules from UML/OCL is presented. Finally, we show how hypotheses with comprehensive predictions may be derived from observations using mixed integer/real programming driven by grammar rules.

Zusammenfassung: Die automatisierte Extraktion hochaufgelöster 3D-Gebäudemodelle aus Bildern und Laserscanner-Daten erfordert starke Modelle für alle Objekte, die im großmaßstäbigen Bereich beobachtbar sind. In diesem Beitrag konzentrieren wir uns auf die Entwicklung eines semantischen Modells für Treppen. Sie spielen eine wichtige Rolle beim Übergang vom Gebäude zum umgebenden Gelände und der Infrastruktur. Diesen Übergangsbereich von Gelände zu Gebäude bezeichnen wir als Gebäudekragen und betrachten Treppen in eben diesem Gebäudekragen. Einfache und komplexe Treppen werden durch UML-Klassendiagramme repräsentiert, wobei semantische und funktionale Aspekte in OCL ausgedrückt werden. Es wird die systematische Ableitung einer attributierten Grammatik mit ihren Produktionsregeln und semantischen Regeln aus UML/OCL vorgestellt. Schließlich wird gezeigt, wie aus Beobachtungen Hypothesen mit umfassenden Prädiktionen unter Verwendung von Methoden des Mixed Integer/Real Programming und gesteuert durch Grammatikregeln konstruiert werden.

1 Introduction

3D building models have been developed since more than a decade, now being readily available for everybody through the Internet. Google Earth, Microsoft's Virtual Earth and NASA World Wind provide high visibility. As a rule, however, existing building models have a low level of detail. Buildings are represented as boxes without roofs and substructures. File formats, such as KML (Keyhole Mark-up Language) used by Google Earth, are poor. Neither topology nor semantics are available. Advanced applications such as planning, disaster management, escape route findings and integrity preserving transactions require a higher level of detail and an explicit representation of semantics in a formal language.

Our research focuses on the design of models, methods and tools for the semi-automatic, interactive refinement of 3D city models. Starting from models consisting of simple blocks and roof structures, the aim is to identify and reconstruct stairs, balconies, windows, doors, and arcades from terrestrial images or laser scans. It is rather obvious to start with entrance stairs. Due to their regular, recursive structure, high-resolution and detailed geometry they may be specified by a handful of parameters. Their semantic relevance is high. An entrance stair connects a route with the entrance door, a building with the outside world. Its detection and reconstruction helps to identify the entrance door and the ground floor level of a building which is not directly observable.

Building models and digital terrain models (DTM) often stem from different sources. This leads to problems of interoperability and homogeneity. A *ground edge*, if given, helps to glue DTM and building together. If there is no exact match, the fiction of a ground edge helps to find an optimal graduation. On a low level of detail a ground edge may be presupposed, although, due to occlusion, it may

not be visible. On a higher level of detail a ground edge may never exist. Instead, there could be a transition zone with intermediate objects such as arcades, light wells etc. We call this transition area the *building collar* (SCHMITTWILKEN et al. 2006).

The aim of this paper is to provide a semantic model of stairs as part of a building collar especially exposing the path from the semantic model to a model usable for data interpretation. It should enable strong hypotheses and predictions based on a small number of noisy and incomplete observations (3D blobs from images or laser range data).

After introducing the main concepts in Section 2 our starting point is the ontology of stairs and a formal specification of object classes, generalization, aggregation and constraint rules defined by the Unified Modelling Language UML and the Object Constraint Language OCL (Section 3). The UML-OCL model allows deciding whether or not a given object is a valid stair but it can't be used to generate a valid stair. Just for interpreting image data using hypotheses-test mode a generative model is needed. For this an attributed context-free grammar is introduced in Section 4. By translating aggregation into (right linear) recursion and OCL rules into semantic rules we show how the UML-OCL model is mapped to an attributed grammar. Whereas a grammar is able to generate all valid sentences of a language, it also may identify the syntactical and – if attributed – semantical structure of a given sentence. The latter is called parsing. Here the problem, however, is not to parse a complete sentence with fixed symbols but to generate reasonable hypotheses from incomplete, noisy observations. In Section 5 we illustrate how hypotheses with comprehensive predictions are derived from observations using thresholds as a bound on distributions from a database of real stairs and mixed integer/real programming. In Section 6 the paper finishes with a discussion on the genericity of the model and future work.

2 Ontology and grammars

Semantics deals with the meaning of words. Ontology deals with the essence of things. Having its roots in philosophy and metaphysics, Plato and Aristotle, Hegel and Kant, Husserl and HEIDEGGER ('Sein und Zeit', 1967) are some of the most prominent thinkers having meditated on this topic. Philosophers put questions like: 'What is the essence of the world?' and 'What characterizes an object?' and studying their approach helps us to make the semantics of objects explicit.

Computer science deals with *ontologies* (in plural). Ontologies make semantics explicit in formal symbolic representations. In other words, an ontology is 'a description of the concepts and relationships that can exist for an agent or a community of agents', generally written 'as a set of definitions of formal vocabulary' (GRUBER 1995). They help to design software providing reasonable answers to sensible questions e.g. they are part of the *Semantic Web*. Ontologies can be used to identify meaningful patterns in noisy observations. Ontologies have roots in Artificial Intelligence (knowledge representation, logic, deduction and automated reasoning), database design (semantic data models) and software engineering (object oriented modelling). Nowadays, there are many ontologies (STAAB 2004), description languages (OWL, ANTONIOU 2004), and platforms (Protégé, GENNARI et al. 2003). We use UML and OCL, the *Unified Modelling Language* together with its *Object Constraint Language* (BOOCH et al. 2005, OMG 2007). The reason is pragmatic: We have developed a large database for approx. 5.000.000 buildings of the German capital city Berlin and the German state North Rhine-Westphalia which is based on the semantic data model of CityGML (GROEGER et al. 2006). CityGML is specified in UML and our model of stairs uses and extends CityGML.

UML is a formal specification and graphical notation which supports the definition of classes, attributes and associations (relations), namely aggregation (building of composites from parts) and inheritance (defining subclasses which inherit attributes and methods from the super class refining the latter by additional attributes and constraints). Generalization/specialization allows defining that the general concept of a stair includes linear stairs having a fixed number of steps, composite stairs with one or more landing in between, branching stairs and many other variants. Aggregation describes the relation between a stair and its steps. In order to claim that all steps of a stair have the same shape OCL has to be used. OCL is a subset of first-order predicate logic including predicates, logical operators such as *AND*, *OR*, *NOT*, *IMPLIES* and existential and universal ($\exists$ and $\forall$) quantifiers (OMG 2006).

Generative grammars have been introduced by CHOMSKY (1956, 1959) in order to reconstruct the syntax of sentences in formal or natural languages. *Context-free* (type-2) languages have a special importance. A context-free language is given by a start symbol S, a set of non-terminals denoted by uppercase letters, a set of terminals denoted by lowercase letters and a set of production rules. Production rules have the form $A \rightarrow \alpha$ where A is a non-terminal and α a sequence of terminals and/or non-terminals. It reads: Each occurrence of the symbol A may be substituted by the string α.

Stairs are taken as an example and denoted as follows: stair by S, riser by R and tread by T. The start symbol S, the set of non-terminals N={S,R,T}, the set of terminals T={r, t} and the following set of production rules are given:

P={ S → R T S → R T S
  R → r T → t}

A typical sentence of this language would be '$rt\ rt\ rt\ rt\ rt = (rt)^5$' denoting a stair with five steps ('rt's).

Context-free grammars define context-free languages. They are expressive enough to define regular, recursive structures such as $\{a^n b^n | n>0\}$. Where a^n stands for n occurrences of a. It is generated by the rules A→ab and A→aAb. The string $a^n b^n c^n$, however cannot be generated by a context-free grammar. An informal reason is 'the number of occurrences of a and c is the same' is context-sensitive and not context-free. This deals with remote substrings. A technical argument using *pumping lemma* is given by HOPCROFT et al. (2001). An important consequence for us is the statement 'all steps of the same stair have the same size' is not context-free.

KNUTH (1968, 1971) proposed the concept of *attribute grammars* which extend context-free grammars to remedy these deficiencies. Terminals and non-terminals are augmented by *attributes* (variables) and production rules by *semantic rules* which specify constraints between attributes. In our example semantic attributes could be height and depth. Semantic rules for the production S→RTS′ could be S.depth=R.depth and S.depth=S′.depth. Note that an apostrophe is used to differentiate between occurrences of the same symbol. A more elaborated attribute grammar for stairs will be given in Section 4.

Attribute grammars are applied in compiler design (AHO et al. 1985). Attributes are used to link remote occurrences of the same symbol, for instance the declaration and instantiation of a variable. Semantic rules are used to generate look-up tables for variable declaration and the generation of fragments of the target code language. Attribute grammars differentiate between *synthesized* and *inherited* attributes in order to specify the flow of information (from right to left or from left to right in a production rule) and the evaluation strategy in a derivation tree. As can be seen later, in our case the flow of information cannot and should not be fixed in advance. On the one hand observation resp. estimation of parameters leads to parameters of a stair, but on the other hand these stair parameters give prediction of unobserved steps and their parameters. Parsing of sentences of a formal language and classification and object reconstruction in noisy images are as a rule rather different. Parsing recognizes the syntactic structure of complete sentences with safely identified non-terminals. Our aim is to deduce reasonable, comprehensive hypotheses and predictions from incomplete observation in noisy images. Attribute grammars and their variants have been used in computer vision and pattern recognition since more than two decades (FU 1982). More recent work on their application in building reconstruction is reported in BRAUN et al. (1995) and STEINHAGE (1999).

Probabilistic (or stochastic) grammars are used to control the generation of words by known a priori probabilities. A stochastic grammar is a grammar where the production rules are associated with probabilities. The probability of a whole derivation equals the product of the probabilities of the single productions of that derivation. The probabilities of all productions $A \rightarrow \alpha$ starting with the same non-terminal A sum up to 1 (FU 1982).

In shape grammars (STINY & GIPS 1972) terminals and non-terminals are geometrical patterns. They have been used to create designs with symmetric, recursive patterns. Shape grammars are appealing because shaped production rules address the intuition of the designer.

For our purpose the main disadvantage of shape grammars is that regularities just have geometrical but no formal symbolic representations. It is more a sketch than a specification. Dependencies and constraints remain implicit. Therefore we prefer attribute (probabilistic) grammars which are as expressive as shape grammars but represent constraints explicitly. In our grammar shapes are represented by attributes and constraints on them. This will be shown in Section 4. Attribute grammars nicely fit to sophisticated constraint solvers (HOOKER 2006).

Research on object modelling using stochastic attribute grammars has been done by MÜLLER et al. (2006) and PARISH & MÜLLER (2001). They use L-Systems, set grammars, and split grammars to generate virtual scenes. However they do not deal with reconstruction from images or 3D point clouds. WONKA et al. (2003) also generate virtual models using split grammars. BRENNER & RIPPERDA (2006) and RIPPERDA & BRENNER (2006) use rjMCMC techniques to control the generation of the derivation tree. MAYER & HUANG (in this issue) use L-Systems and MCMC techniques to generate 3D hypotheses for trees.

3 Ontology of stairs

Stairs are designed to allow pedestrians to surmount the altitude difference of several levels. Contrary to ladders on one hand and ramps on the other hand stairs combine the comfort of walking with saving of space. Stairs are regularly formed from steps having the same rise and depth. Sometimes they are discontinued by one or more landings. The rise of a riser and depth of a tread are adapted to the step length of adult humans as BLONDEL already assumed in 1698. The vocabulary of stairs and parts of the taxonomy are defined in ISO 3881 (ISO 1977) to which the following specification adheres.

It is a general observation in ontological engineering that there are several ontologies for the same class of objects. An ontology refers to a domain of discourse, a cognitive interest and a greater context. It is difficult in general to match different ontologies for the same entities, but this discussion is beyond the scope of this paper. FRANK (1996, 2003) and KUHN (2005) advice to identify the actions to which the ontology refers. For us two perspectives and two actions are relevant: The pedestrian has to cover an altitude difference. To this end he uses a stair. His action is stepping. A completely different perspective starts with terrestrial images or laser scans and ends up with the reconstruction of a stair. The action is classification and object reconstruction. Both kinds of action include that the visible surface plays an important role – walking on the surface on one hand and its observation and reconstruction on the other hand. The visible surface is associated with meaningful objects. Stairs are specified in ISO 3881 (cf. Figure 1). In this norm the basic terms of stairs are standardized, their meaning is defined, and their translation into German and French is fixed. ISO 3881 provides a thesaurus.

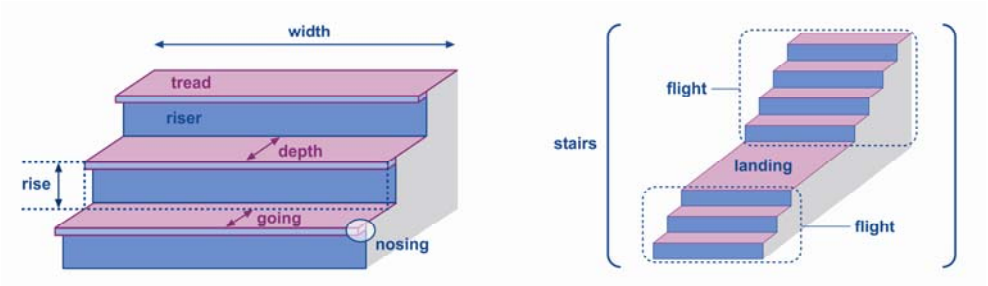

Figure 1: Basic terms of stairs defined by ISO 3881

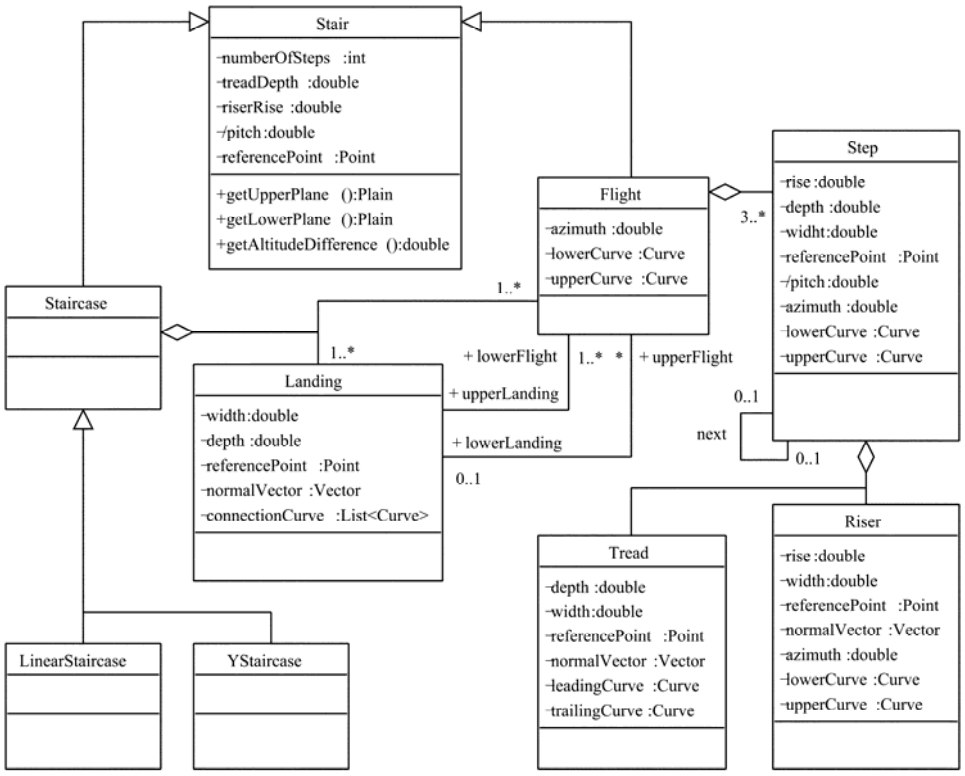

Figure 2: UML diagram of stairs as one part of an ontology

The UML diagram shown in Figure 2 provides a semantic model of stais.. It gives a general overview of stairs, its parts and their dependencies. On the top level there is the abstract class `Stair (S)` which is specialised into the classes `Staircase` and `Flight (F)`. Since attributes should be observable `pitch` is marked as a *derived* attribute. The class `Step` is aggregated by one `Tread` and one `Riser`. As mentioned above tread and riser are the basic components of a stair. All `referencePoint` attributes are dependent on each other due to the recursive and regular structure of stairs. The step class also has an association `next` with itself. So each step points to its successor (in upward direction). Three or more steps aggregate to a stair. Here we follow the German norm DIN 18065 (DIN 2000). Single and double steps have rather different patterns. A flight is exclusively made of steps. A staircase is an aggregation of flights and landings. Two associations express which flights are upstairs from the landing (optional) and which are downstairs (at least one flight). A landing has to have at least one lower flight and optional upper flights. Furthermore `LinearStaircase (L)` and `YStaircase (Y)` are special staircases and hence special stairs. A linear staircase is introduced as a staircase with one lower flight and one upper flight per landing and a Y-staircase as an aggregation of a landing with two lower flights and one upper flight or vice versa. In the following it is assumed that each flight has a constant width and a straight ground plan.

Whereas classes, attributes association, aggregation and inheritance structures are represented in the UML diagram as described above, essential regularities which characterize the essence of stairs are not given yet. They are specified by object constraint rules. Some basic assumptions and observations on stairs in general and thus on entrance stairs are the following ones, given in natural language. The corresponding lines of the transcribed OCL expressions shown in Table 1 are given in brackets.

1. All steps of a stair have the same rise and tread. This holds both for linear staircases with or without landing and for Y-staircases (lines 4-5).

2. Subsequent steps are connected seamlessly, i.e. the succeeding step follows the preceding step immediately (line 8).
3. The altitude difference between the two ends of a stair is given by the number of steps times the rise (line 9).
4. Tread and rise specify a vertical angle called pitch (line 3). There is another horizontal angle, the azimuth (line 3), which is the same for the steps of a flight (line 6), which may or may not be different for flights connected by a landing and which is different for y-staircases (lines 10-12). In all cases, there is a defined relationship between the azimuth and the orientation of the front façade of the corresponding house.
5. Since stairs are for human adults, depth and rise both have a given range. On the basis of a database containing parameters of 120 entrance stairs, one may very well (a) assume the rise, the tread depth, and the footstep of different stairs to be normally distributed (cf. lines 1, 2, and 9) and (b) observe the footstep formula 2r + d as the most stable parameter for stairs (line 7). Average values and standard deviations for the observed parameters are given in Table 2.

The OCL rules are given in Table 1. For the sake of readability the notation is slightly modified for this example. Each rule defines an **inv**ariant in the **context** of a specific class.

```
01  context Riser inv:  rise ~ N(17.0, (1.2)²)
02  context Tread inv: depth ~ N(30.8, (2.7)²)
03  context Step inv:   pitch = riser.rise / tread.depth & azimuth = riser.normalVector
04  context Flight inv: step → ∀ s1, s2 : s1.tread.depth = s2.tread.depth &
05                      step → ∀ s1, s2 : s1.riser.rise = s2.riser.rise &
06                      step → ∀ s1, s2 : s1.azimuth = s2.azimuth &
07                      step → ∀ s : (2 * s.rise + s.tread) ~  N (62.6, (3.0)²) &
08                      step → ∀ s : s.upperCurve = s.next.lowerCurve
09  context Staircase inv: getAltitudeDifference() = sum(flight.riser.rise) * flight.riser->size() &
10                      flight → ∀ f1, f2 :  abs(f1.azimuth - f2.azimuth) = 0 or
11                                           abs(f1.azimuth - f2.azimuth) = π / 2 or
12                                           abs(f1.azimuth - f2.azimuth) = π
```

Table 1: OCL rules expand the UML diagram with constraints

	riser [cm]	**tread** [cm]	**r / t** [ ]	**2r + t** [cm]
μ	17.0	30.8	0.6	62.6
σ	1.2	2.7	0.1	3.0
μ / σ [ ]	0.07	0.09	0.13	0.05

Table 2: Parameter distribution on staircases

4 From ontology to grammar

Table 3 gives the production rules corresponding to the UML diagram shown in Figure 2. The production rules are derived from the UML diagram by a method which consists of the following rules plus some technical details.

Each class name is associated with a non-terminal symbol. Non-terminals are given by the set {S,L,Y,F,LANDING,STEP,RISER,TREAD}. For the sake of brevity the non-terminals for stair (S), linear staircase (L), Y-staircase (Y), and flight (F) are abbreviated. The rest of them will drop out as can be seen in the following. All non-terminals have a one-to-one correspondence with the classes of the UML diagram.

The terminals are in the set `{riser,tread,landing}`. They correspond to – but are not identical with – features which are directly observable. The main point here is that the corresponding classes of the terminals are primitive in a specific sense: delete from the UML 'graph' all edges which are neither aggregation nor generalization. Give the generalization edges a direction from super class to subclass and the aggregation edges a direction from whole to part. If a super class defines an aggregated object, add the corresponding edges to their subclasses, too. The primitive classes are just those nodes of the resulting directed graph which have no off-going edges. Thus one gets the primitive classes `Landing`, `Tread` and `Riser`. These primitive classes form the set of terminal symbols. Note that `L` and `Y` are no leaves since they participate in the aggregation of `Staircase`.

As the three rules for stair illustrate, generalization is easy to handle: Translate generalization to a set of rules, where the super class non-terminal is on the left hand side and each subclass non-terminal gives a right hand side.

Aggregation is more difficult to handle, but there are two important patterns. The first case is where the whole (step) is composed from two or more different parts (tread and riser) the number of which is given in advance. Then the part non-terminals form the right hand side and the aggregate forms the left hand side of the corresponding rule. Second case: if there is a non-fixed number of parts, as it is the case with flight and steps, recursion is applied. As the case of flight shows, one linear *right recursive* rule

(`F → STEP F`)

and one non-recursive rule

(`F → STEP`)

suffice. A recursive production rule is called *linear recursive*, if there is only one occurrence of the left hand symbol on the right hand side of the production. It is called *right recursive*, if the recursive symbol is right-most. This is an easy case of recursion since it forms the special case of a regular (Chomsky 3) grammar. The case of landing is similar.

The derived grammar makes the immediate correspondence to the UML diagram explicit but it has some redundancies which can easily be removed by rewriting. The three production rules for landing, tread, and riser just have one non-terminal symbol on the left and one terminal on the right hand side. When all occurrences of these non-terminal symbols in the remaining productions are replaced by the terminals, both the three production rules and the non-terminals can be cancelled.

There are three additional terminal symbols which correspond to no class, namely '`;`', '`(`', and '`)`'. These are just meta-symbols which are used in the case of Y-staircases in order to make a non-linear tree-like structure explicit. They support readability, but do not provide additional information.

p1: **S** → **L** | **Y** | **F**
p2: **L** → **F** landing **L** | **F**
p3: **Y** → **F** landing (**F** ; **F**) | (**F** ; **F**) landing **F**
p4: **F** → riser tread **F** | riser tread

Table 3: Excerpt of the grammar derived from UML diagram

So far only classes, aggregation and generalization have been reflected. Other associations may be mapped as well provided that their multiplicity is given and their precise semantics are specified by constraint rules. This, however, is beyond the scope of this paper. What about the *attributes*? They become attributes of the respective terminals and non-terminals. Table 4 shows some examples of semantic rules. (Apostrophe is used to differentiate between multiple occurrences of the same symbol in one production. Non-scalar attributes are underlined.) Following this pattern the other semantic rules may easily be derived from the OCL rules. A comprehensive list of OCL rules (as well as detailed production rules, attributes and semantic rules) can be found at http://www.ikg.uni-bonn.de/fileadmin/data/schmittwilken_07_modeling_appendix.pdf. Although the correspondence between semantic rules and OCL rules is rather obvious, the translation is a bit tricky. Firstly, note

that aggregation with identities on a non-fixed number of parts (flight and step) affords a quantifier on all steps `s1` and `s2` in the OCL rule. In the production rule, aggregation is represented by (right linear) recursion, and in the semantic rule, there are dependencies between the attributes of the two recursive occurrences of flight and the non-recursive non-terminal riser and tread. Secondly, inaccuracy is represented differently. OCL rules define constraints for the strict ontological model. For example they ensure that all steps have the same attributes and they provide parameters (mean μ and standard deviation σ) of the respective distributions. On the other hand there are the semantic rules of the attribute grammar. They are used in an operational scenario in order to guarantee a given functional context for each attribute e.g. the attributes of a specific flight equal the weighted mean of the respective attributes of its steps. The exact stochastic definition of the functions *weightedMean()* and *f()* is not precisely specified here.

F → riser tread F'
F.numberOfSteps = F'.numberOfSteps + 1
F.rise = weightedMean(riser.rise, F'.rise)
F.azimuth = f(riser.normalVector, F'.azimuth)
tread.referencePoint = sum(riser.referencePoint, dx, dy)
... (e.g. topological connection of riser and tread)

Table 4: Excerpt of the semantic rules for the unknown attributes

Thresholds may be derived from the deviation (f.i. $\Delta = 3\sigma$) and used to define boolean-valued semantic rules called *guards.* Guards control the search and allow to prune the search space for candidate models early. Examples are given in Table 5 referring to a (simplified version of) production rule p2 in Table 3. *Guarded semantic rules* mimic *Guarded Horn Clauses* described in UEDA (1985). The *guard* of a production rule is the conjunction of all guarded semantic rules. An application of a production is *valid* if its guard evaluates true. A *derivation* is *valid* if all applications of production rules are valid, and a sentence is valid if it has a valid derivation. Formally, the valid sentences form a subset of the syntactically correct sentences specified by the corresponding non-attribute grammar.

L → F landing F'
| F.treadDepth – F'.treadDepth | < 8.02
| F.rise – F'.rise | < 3.57
| F.pitch – F'.pitch | < 0.22
...

Table 5: Guard rules for the unknown attributes

5 Hypotheses by solving mixed linear constraints

In this paragraph it is illustrated how the semantic rules and the guard rules may be used to derive hypotheses being consistent with given observations. To simplify illustrations we study a 2D projection of stairs. Generalization to 3D is obvious.

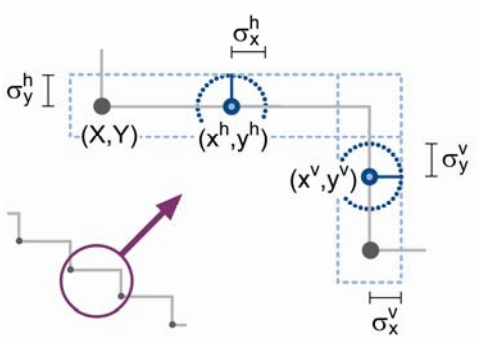

Figure 3: Variables used for the reasoning example.

Assume that for the surfaces of stairs we have observed 2D points and their accuracies distinguished from points on horizontal and vertical faces (c.f. Figure 3). Each point 'knows' his direction i.e. the normal vector of 'his' face, so we call this points *2D needles*.

$$x_i^h, y_i^h, \sigma_x^h, \sigma_y^h, x_i^v, y_i^v, \sigma_x^v, \sigma_y^v \quad (1)$$

The major construction principle of stairs is the constancy of rise and tread depth within a stair. So we claim

$$r = const \wedge t = const \quad (2)$$

Thresholds for these values can be derived from the stair database mentioned in section 3:

$$14 \le r \le 20 \wedge 18 \le t \le 38 \quad (3)$$

The number of steps within a stair is an integer number, i.e. the number of treads (n) and the number of risers (m):

$$n \in N \wedge m \in N \quad (4)$$

The structure of a 2D stair with origin (0,0) is defined by its rise and tread depth apart from its number of steps. So *reference points* can be introduced with coordinates resulting from a multiple of rise or tread depth:

$$Y_n = Y_0 + m * r \wedge X_n = X_0 + n * t \quad (5)$$

The coordinates of observed points supporting a given set of stair parameters can be specified by the following constraints. Inequalities of 'vertical points' can be derived analogously.

$$\begin{aligned} X_n - \sigma_x^h \le x_i^h \le X_n + \sigma_x^h + t \\ Y_n - \sigma_y^h \le y_i^h \le Y_n + \sigma_y^h \end{aligned} \quad (6)$$

We give the chain of reasoning which is used to determine the parameters of a stair from given observations. We assume that the observations are given within the stair coordinate system. Without loss of generality, the origin of the stair is claimed to be $(X_0, Y_0) = (0,0)$ and all accuracies are assumed to be 1.

Which kind of solver is necessary in order to support and implement this kind of reasoning?

Based on the ontology of stairs, constraints defined in OCL and the semantic rules of the attribute grammar turn out to become linear equalities and inequalities. The products $m*r$ and $n*t$ will disappear by constraint propagation and/or tentative instantiations of n and m. Thus a solver for linear inequalities (where equalities are a special case) is needed. The method of choice would be Simplex, but it does not suffice in our case. Simplex assumes real valued variables but there are integer variables as well (number of steps). Linear optimization on integers is computationally more demanding

than optimization on real numbers. Optimization here becomes combinatorial and in fact NP-complete. More specifically, the given problem is mixed-integer. Fortunately, there are algorithms and solvers for this kind of constraints (HOOKER 2006).

We use the constraints solver ECLiPSe (APT & WALLACE, 2007) to estimate integer values or real intervals for the interdependent parameters. All constraints have been implemented. Table 6 shows an excerpt of the programm: two clauses for estimating the reference point from an observed point lying on a horizontal face.

```
estimate_RefX_from_hPoint( ) :-
  x $>= RefX - Sigma_x_horizontal,
  x $=< RefX + Sigma _x_horizontal + Tread,
  RefX $= X0 + N * Tread.

estimate_RefY_from_hPoint( ) :-
  y $>= RefY - Sigma _y_vertical,
  y $=< RefY + Sigma _y_vertical,
  RefY $= Y0 + M * Riser.
```

Table 6: Excerpt of the constraint solver program

Starting with the observation *p1=(22,0)* we cannot reduce the intervals for riser and tread depth: *tread* $\in$ *[18,38]* and *riser* $\in$ *[14,20]*. Obviously we can fix the coordinates of the origin as *(0,0)*. Insertion of the observation values into (7) we get

$$22-1-[18..38] \le X_1 \le 22+1$$

$$0-1 \le Y_1 \le 0+1$$

and due to (2-5) we get $X_1 \in [0,23]$, and $Y_1 \in [0,1]$. This leads to the number of risers and treads (from origin to reference point): $n1 \in [0,1]$ and $m = 0$.

The second observation *p2=(140,75)* restricts the domain but the parameters are still ambiguous: *tread* $\in$ *[26,38]* and *riser* $\in$ *[18.3,19]*. So we estimate for the second reference point $X2 \in [101,137]$, and $Y2 \in [74,76]$ with $n2 \in [3,5]$ and $m2 = 4$. The intervals of the coordinates of the origin and the first reference point remain the same.

The third observation *p3=(71,26)* will resolve the ambiguity: *tread* $\in$ *[31,36]* and *riser* $\in$ *[18.3,19]* with the following numbers of steps: *n1=0, m1=0, n2=3, m2=4, n3=2*, and *m3=6.*

If we get a fourth observation *such as p4=(291,115)* we can even detect landings within the stair. The estimation of the fourth reference point and its numbers of steps from the origion leads to an inconsitency: $n4 \in [8,9]$ and $m4=6$. The difference between n4 and m4 can only be explained by a landing between p2 and p3.

6 Conclusion and final remarks

We have specified an ontology of stairs in the building collar based on 'common sense' and ISO 3881. The UML/OCL model presented in Section 3 and the attribute grammar of Section 4 specify a subset of the variety of stairs which is observed in reality. How to derive the UML diagram and the constraint rules from examples automatically remains an open question. We have sketched a procedure which maps the UML diagram and the constraint rules to an attribute grammar. With regard to associations the focus was on aggregation and generalization. The mapping of other associations depends on their multiplicity and their semantics specified by OCL rules.

One main contribution of this paper is the derivation of logical formulas (conjunctions of constraints) from the ontology of stairs linked by the (stochastic) attribute grammars. We provide a method to solve these constraints and generate candidate models. This method has been implemented as a prototype in ECLiPSe and is available on the internet at http://www.ikg.uni-

bonn.de/fileadmin/data/schmittwilken_07_modeling_appendix.pdf. The candidate models fix numbers of integral domains and specify precise intervals for the real domains. The stochastic estimation of most probable values within these intervals with Bayes statistic (RUSSEL & NORVIG 2003, BISHOP 2006, PEARL 2000) is left as a topic for future research.

The constraints solver generates candidate models from consistent observations. A set of observations is consistent if there exists an instantiation of the model parameters satisfying the constraints. These parameters constitute a model explaining the observations. Under certain conditions three points (3D needles) suffice to specify a single candidate model. Thus the constraint solver is able to handle incomplete observations. In realistic scenarios, however there will be a large number of 3D points with substantial percentage of outliers. The reconstruction of stairs from noisy observations requires the differentiation between inliers and outliers. From a logical point of view inliers constitute the largest consistent subset of the given observations. Exact algorithms for the identification of this subset are not feasible. An obvious probabilistic approach to identify this subset is the RANSAC algorithm (FISCHLER & BOLLES 1981) which nicely fits to our constraint solver and will b implemented in a next step. For a randomly chosen minimal set of observations the constraint solver derives one or more candidate models if this set of observations is consistent. In a next step RANSAC calculates the 'consensus set' of consistent observations which fits into the candidate model. The RANSAC algorithm iterates its steps and terminates when a sufficiently large set has been found.

Although we focus on entrance stairs of 'ordinary' buildings which are less complex than those to be seen in pleasure grounds, natural parks and in front of castles, some restrictions should be mentioned. We assume that stairs have constant width. In some cases the width decreases (e.g. parallelogram layout), and it is a linear function of the index of the step. We also assume that stairs are straight, branching (Y-stairs) or interrupted by landings. Some stairs have more sophisticated, curved layouts. Another assumption is that steps are compact solids. However some steps have nosing, others rest on piles (open stairs). Reconstruction and classification of the mentioned details is more demanding and requires an extension of the ontology and thus additional model parameters.

An important subclass of stairs is spiral stairs. They have limited relevance for building collars , fire escapes and indoor staircases, however, sometimes have this shape. Whereas riser and number of steps can be estimated by the constraints given in Section 5 the tread depth and the width of the stair are more sophisticated and require trigonometric functions in a Cartesian coordinate system. Linearity can be achieved in a polar coordinate system. But this affords that the centre point(s) of the spiral stair can be derived Curved transitions between two flights have a similar pattern. An effective system of constraints for this case is topic for future research.

The ontology and the attribute grammar are generic. Extensions for the cases mentioned above can be handled in this framework. They require additional parameters and more sophisticated constraints. Above, they require non linear constraint solvers. However there are algorithms (BOYD & VANDENBERGHE 2006) and systems like CPLEX (ILOG 2006), ECLiPSe (APT & WALLACE, 2007), or Frontline solver (FRONTLINE SYSTEMS, INC., 2007) which are able to handle large numbers of non-linear constraints if they are convex. Even non-convex collections of constraints can be solved if they meet the requirement that the modelling of the independencies of the constraints supports constraint propagation. In Section 5 we have shown that the non-linearity of $n * tread$ disappears when the domain of n becomes a singleton due to constraint propagation. Thus we claim that the integration of strong constraint solvers with graphical models / Bayes networks will help to bridge the gap between ontological engineering and 3D object reconstruction.

References

AHO, A. V., SETHI, R. & ULLMANN, J. D., 1985: Compilers. Principles, Techniques and Tools. – Addison-Wesley Longman, Boston, USA.

ANTONIOU, G. & VAN HARMELEN, F., 2004: Web Ontology Language OWL. Handbook on Ontologies. – Springer, New York, USA: 68–92.

APT, K. R. & WALLACE, M., 2007: Constraint Logic Programming Using ECLiPSe. – Cambridge University Press, Cambridge, UK.

BISHOP, C. M., 2006: Pattern Recognition and Machine Learning. – Springer, New York, USA.

BLONDEL, F., 1698: Cours D'Architecture. – L'Academie Royale D'Architecture, Paris, France.

BOOCH, G., RUMBAUGH, J. & JACOBSON, I., 2005: Unified Modeling Language User Guide. – Addison-Wesley Longman, Boston, USA.

BOYD, S. & VANDENBERGHE, L., 2006: Convex Optimization. – Cambridge University Press, Cambridge, UK.

BRAUN, C., KOLBE, T. H., LANG, F., SCHICKLER, W., STEINHAGE, V., CREMERS, A. B., FÖRSTNER, W. & PLÜMER, L., 1995: Models for Photogrammetric Building Reconstruction – Computers & Graphics **19** (1): 109–118.

BRENNER, C., & RIPPERDA, N., 2006: Extraction of Facades using rjMCMC and Constraint Equations – The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences **36**(3): 155–160.

CHOMSKY, N., 1956: Three Models for the Description of Language. – IEEE Transactions on Information Theory **2** (3): 113–124.

CHOMSKY, N., 1959: On Certain Formal Properties of Grammars. – Information and Control **2**: 137–167.

DIN (DEUTSCHES INSTITUT FÜR NORMUNG), 2000: DIN 18065 Gebäudetreppen – Definitionen, Meßregeln, Hauptmaße – Berlin, Germany.

FISCHLER, M. A. & BOLLES, R. C., 1981: Random Sample Consensus: a Paradigm for Model Fitting With Application to Image Analysis and Automated Cartography – Communications of the ACM, **24**(6): 381–395.

FRANK, A.U., 1996: Ontology: A Consumer's Point of View, Spatial and Temporal Reasoning. – Kluwer, Dordrecht, Netherlands.

FRANK, A.U., 2003: Ontology for Spatio-temporal Databases. – In: KOUBARAKIS, M., SELLIS T.K., FRANK, A.U. et al. (eds.): Spatio-Temporal Databases: The CHOROCHRONOS Approach. Lecture Notes in Computer Science 2520, Springer, Berlin, Germany: 9–78.

FRONTLINE SYSTEMS, INC., 2007: website. http://www.solver.com (last visited: 2007-06-20).

FU, K. S., 1982: Syntactic Pattern Recognition and Application. – Prentice Hall, Englewood Cliffs, USA.

GENNARI, J., MUSEN, M. A., FERGERSON, R. W., GROSSO, W. E., CRUBEZY, M., ERIKSSON, H., NOY, N. F. & TU, S. W., 2003: The Evolution of Protégé: An Environment for Knowledge-Based Systems Development – International Journal of Human-Computer Studies, **58**(1), 89–123.

GRÖGER, G., KOLBE, T. H. & CZERWINSKI, A., 2006: Candidate OpenGIS CityGML Implementation Specification (City Geography Markup Language). – OGC Doc. No. 06–057r1.

GRUBER, T., 1995: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. – International Journal of Human-Computer Studies **43** (4–5): 907–928.

HEIDEGGER, M., 1967: Sein und Zeit. – Niemeyer, Tübingen, Germany.

HOOKER, J. N., 2006: Operations Research Methods in Constraint Programming. – In: ROSSI, F., VAN BEEK, P. & WALSH, T. (eds.): Handbook of Constraint Programming. Elsevier, Amsterdam, Netherlands: 527–570.

HOPCROFT, J. E., MOTWANI, R. & ULLMAN, J. D., 2001: Introduction to Automata Theory, Languages, and Computation. – Addison-Wesley Longman, Amsterdam, Netherlands.

HUANG, H. & MAYER, H., 2007: Extraction of the 3D Branching Structure of Unfoliaged Deciduous Trees from Image Sequences. – Accepted for Photogrammetrie – Fernerkundung – Geoinformation (6/2007).

ILOG, 2006: Efficient modelling in ILOG OPL-CPLEX Development System. – White Paper, http://www.ilog.com/products/oplstudio/whitepapers/index.cfm (last visited 2007-05-15).

ISO (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION), 1977: Building construction - Stairs - Vocabulary - Part I. – ISO 3880/I-1977, First Edition, Geneva, Switzerland.

KNUTH, D. E., 1968: Semantics of Context-Free Languages. – Theory of Computing Systems **2** (2): 127–145.

KNUTH, D. E., 1971: Top-Down Syntax Analysis. – Acta Informatica **1** (2): 79–110.

KUHN, W., 2005: Why, of What, and How? – In: SPACCAPIETRA, S. & ZIMÁNYI, E. (eds.): Semantic-based Geographical Information Systems. Journal on Data Semantics **III**. Lecture Notes in Computer Science 3534, Springer, Berlin, Germany: 1–24.

MÜLLER, P., WONKA, P., HAEGLER, S., ULMER, A., & GOOL, L. VAN, 2006: Procedural Modeling of Buildings – In: Proceedings of ACM SIGGRAPH 2006, New York, USA / ACM Transactions on Graphics **25**, ACM Press: 614–623.

OMG (OBJECT MANAGEMENT GROUP), 2006: Object Constraint language (OCL) 2.0.

OMG (OBJECT MANAGEMENT GROUP), 2007: Unified Modeling Language (UML) 2.1.1.

PARISH, Y. I. H., & MÜLLER, P, 2001: Procedural Modeling of Cities – In: FIUME, E. (ed.), Proceedings of ACM SIGGRAPH 2001, New York, USA, ACM Press: 301–308.

PEARL, J., 2000: Causality. Models, Reasoning and Inference. – Cambridge University Press, Cambridge, UK.

RUSSELL, S. J., NORVIG, P., 2003: Artificial Intelligence: A Modern Approach. – Prentice Hall, Pearson Education, Upper Saddle River, USA.

RIPPERDA, N., & BRENNER, C., 2006: Reconstruction of facade structures using a formal grammar and rjMCMC – Lecture Notes in Computer Science **4174**: 750–759.

SCHMITTWILKEN, J., KOLBE, T. H. & PLÜMER, L., 2006: Der Gebäudekragen - Eine detaillierte Betrachtung des Übergangs von Gebäude und Gelände. – In: SEYFERT, E. (ed.): Proceedings of 26. Wissenschaftlich-Technische Jahrestagung der DGPF, September 2006, Berlin, Germany: 127–135.

STAAB, S. & STUDER, R., 2004: Handbook on Ontologies. – Springer, Berlin, Germany.

STADLER, A. & KOLBE, T. H., 2007: Spatio-Semantic Coherence in the Integration of 3D City Models. – In: Proceedings of the 5th International Symposium on Spatial Data Quality ISSDQ 2007, June 2007, Enschede, Netherlands, ISPRS Archives (in print).

STEINHAGE, V., 1999: Zur automatischen Gebäuderekonstruktion aus Luftbildern. – Habilitationsschrift an der Mathematisch-Naturwissenschaftlichen Fakultät der Universität Bonn.

STINY, G. & GIPS, J., 1972: Shape Grammars and the Generative Specification of Painting and Sculpture. – In: FREYMAN, C. V. (ed.): Information Processing **71**. North-Holland, Amsterdam, Netherlands: 1460–1465.

UEDA, K., 1985: Guarded Horn Clauses. – In: WADA, E. (ed.): Logic Programming '85: Proceedings of the 4th Conference. Lecture Notes in Computer Science **221**, Springer, New York, USA: 168–179.

WONKA, P., WIMMER, M., SILLION, F., & RIBARSKY, W., 2003: Instant Architecture – ACM Transactions on Graphics, **22**(4), 669–677.

Authors' address:

Dipl.-Ing. JÖRG SCHMITTWILKEN, Prof. Dr. LUTZ PLÜMER, Department of Geoinformation, Institute of Geodesy and Geoinformation, University of Bonn, Meckenheimer Allee 172, 53115 Bonn, Tel.: +49 228/73-{1756, 1751}, E-Mail: {schmittwilken, pluemer}@ikg.uni-bonn.de

Dipl.-Ing. JENS SAATKAMP, Prof. Dr.-Ing. WOLFGANG FÖRSTNER, Department of Photogrammetry, Institute of Geodesy and Geoinformation, University of Bonn, Nussallee 15, 53115 Bonn, Tel.: +49 228/73-{2711, 2713}, E-Mail: saatkamp@uni-bonn.de, wf@ipb.uni-bonn.de

Prof. Dr. THOMAS H. KOLBE, Institute for Geodesy and Geoinformation Science, Technical University Berlin, Straße des 17. Juni 135, 10623 Berlin, Tel.: +49 30/314-23274, E-Mail: kolbe@igg.tu-berlin.de